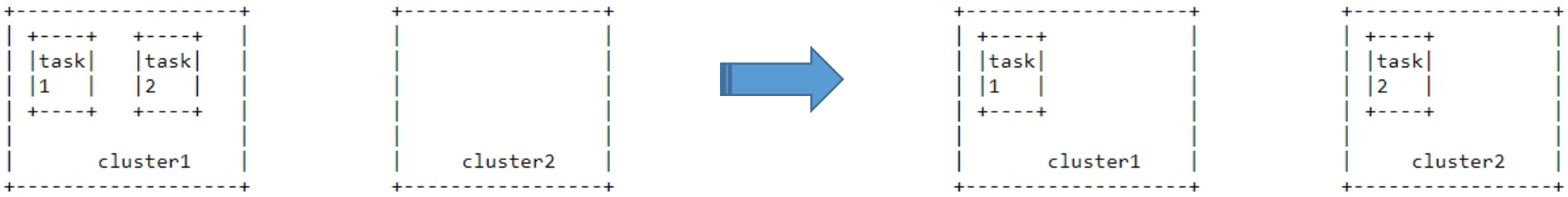
Cluster Scheduler RFC v6

Barry Song

Optimizing tasks deployment

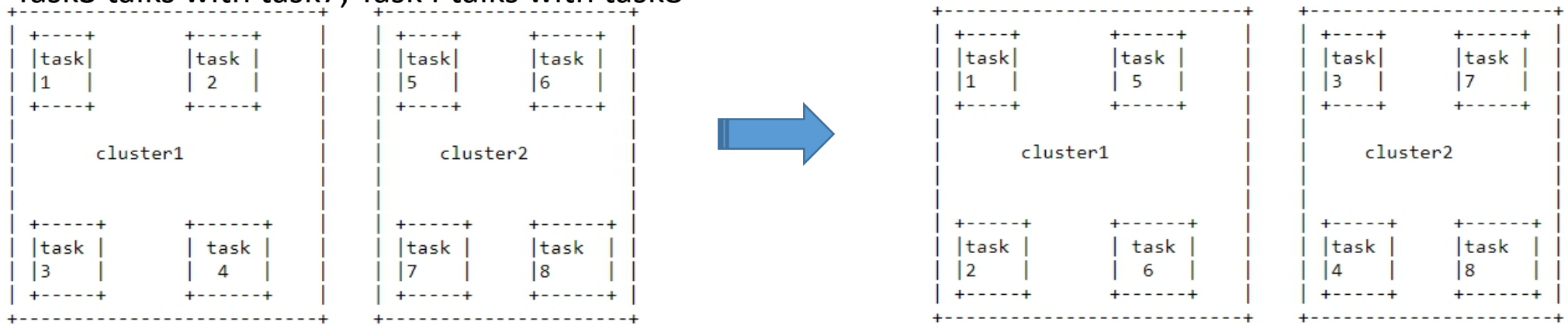
- spread unrelated task

✓ task1 doesn't talk with task2



- pack related task

Task1 talks with task5; Task2 talks with task6;
Task3 talks with task7; Task4 talks with task8



current approach

RFC v6: https://lore.kernel.org/lkml/20210420001844.9116-1-song.bao.hua@hisilicon.com/

- added cluster sched_domain

-> Load Balance will help spread tasks

- changed WAKE_AFFINE

-> if waker and wakee are in the same LLC(NUMA), scan cluster instead of LLC

```
static int
select_task_rq_fair(struct task_struct *p, int prev_cpu, int wake_flags)
{
        int sync = (wake_flags & WF_SYNC) && !(current->flags & PF_EXITING);
        struct sched_domain *tmp, *sd = NULL;
        int cpu = smp_processor_id();
        int new_cpu = prev_cpu;
        int want_affine = 0;
        /* SD_flags and WF_flags share the first nibble */
        int sd_flag = wake_flags & 0xF;
        /*
         * if cpu and prev_cpu share LLC, consider cluster sibling rather
         * than llc. this is typically true while tasks are bound within
         * one numa
         */
        int cluster = sched_cluster_active() && cpus_share_cache(cpu, prev_cpu, 0);

        if (wake_flags & WF_TTWU) {
                record_wakee(p);

                if (sched_energy_enabled()) {
                        new_cpu = find_energy_efficient_cpu(p, prev_cpu);
                        if (new_cpu >= 0)
                                return new_cpu;
                        new_cpu = prev_cpu;
                }
```

-> wake_wide(), select_idle_sibling() will move to use cluster while cluster==true

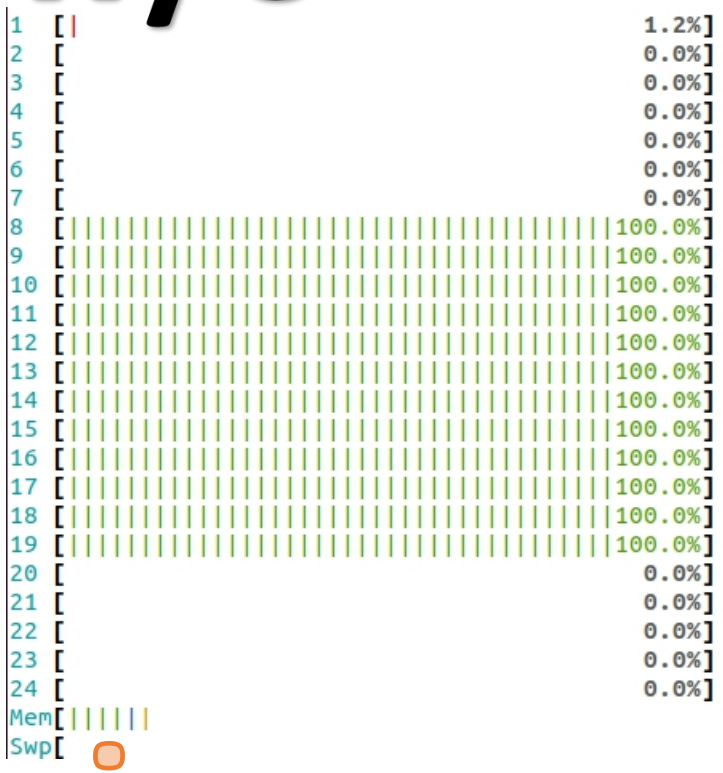
```
bool cpus_share_cache(int this_cpu, int that_cpu, int cluster)
{
        if (cluster)
                return per_cpu(sd_cluster_id, this_cpu) == per_cpu(sd_cluster_id, that_cpu);
        else
                return per_cpu(sd_llc_id, this_cpu) == per_cpu(sd_llc_id, that_cpu);

}
```

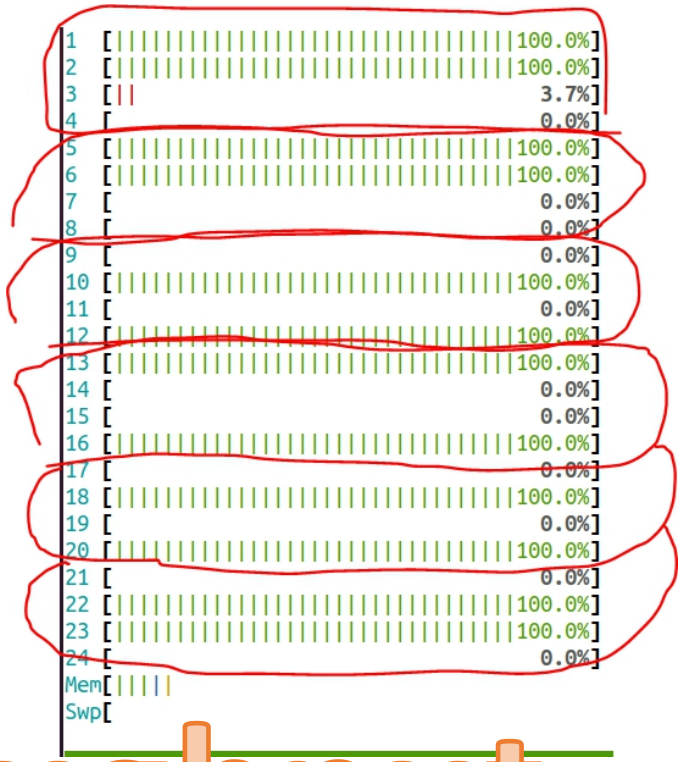
Spread unrelated tasks(demo)

numactl -N 0 /usr/lib/lmbench/bin/stream -P 12 -M 1024M -N 5

w/o



w/



improved throughput

Pack related task(demo)

numactl -N 0 hackbench -p -T -l 1000000000 -f 1 -g 6 (12 threads, 6 sender-receiver couples)

w/o: couples are randomly deployed

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND	P
2347	root	20	0	3580	556	444	R	93.7	0.0	3:02.93	hackbench	13
2346	root	20	0	3580	556	444	R	99.7	0.0	3:15.23	hackbench	12
2345	root	20	0	3580	556	444	R	93.4	0.0	3:02.30	hackbench	15
2344	root	20	0	3580	556	444	R	99.9	0.0	3:15.24	hackbench	10
2343	root	20	0	3580	556	444	R	93.4	0.0	3:07.12	hackbench	9
2342	root	20	0	3580	556	444	R	99.9	0.0	3:15.14	hackbench	8
2341	root	20	0	3580	556	444	R	99.7	0.0	3:15.22	hackbench	3
2340	root	20	0	3580	556	444	R	99.3	0.0	3:14.85	hackbench	5
2339	root	20	0	3580	556	444	R	99.9	0.0	3:13.03	hackbench	14
2338	root	20	0	3580	556	444	R	99.7	0.0	3:15.22	hackbench	4
2337	root	20	0	3580	556	444	R	93.4	0.0	3:02.10	hackbench	1
2336	root	20	0	3580	556	444	R	99.7	0.0	3:15.23	hackbench	0

w: couples are packed

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND	P
2053	root	20	0	3580	504	436	R	99.9	0.0	0:42.36	hackbench	7
2052	root	20	0	3580	504	436	R	99.3	0.0	0:42.18	hackbench	5
2051	root	20	0	3580	504	436	R	93.4	0.0	0:39.69	hackbench	21
2050	root	20	0	3580	504	436	R	99.7	0.0	0:42.35	hackbench	20
2049	root	20	0	3580	504	436	R	99.9	0.0	0:42.36	hackbench	19
2048	root	20	0	3580	504	436	R	99.7	0.0	0:42.16	hackbench	16
2047	root	20	0	3580	504	436	R	99.7	0.0	0:42.36	hackbench	8
2046	root	20	0	3580	504	436	R	99.0	0.0	0:42.02	hackbench	9
2045	root	20	0	3580	504	436	R	99.7	0.0	0:42.36	hackbench	1
2044	root	20	0	3580	504	436	R	99.3	0.0	0:42.22	hackbench	2
2043	root	20	0	3580	504	436	R	99.9	0.0	0:42.36	hackbench	15
2042	root	20	0	3580	504	436	R	99.9	0.0	0:42.28	hackbench	12

decreased latency